CLAIM LISTING

1. (Previously presented) A server system comprising:

one or more computers; and

a multi-layer application executing on the computers to handle client requests submitted by various client devices, the multi-layer application comprising:

a problem-solving logic layer to process the client requests according to an associated problem domain, wherein the problem domain pertains to a particular category of service, the problem-solving logic layer containing one or more execution models to perform various sets of tasks when processing the client requests, the problem-solving logic layer producing replies to the client requests;

an execution environment layer to receive the client requests and select an appropriate execution model in the problem-solving logic layer for processing the client requests;

an interfacing layer to interface the problem-solving logic layer with one or more resources so that the execution models may utilize the resources when processing the client requests; and

a presentation layer to receive the replies produced by the problem-solving logic layer and to structure the replies in a manner that makes the replies presentable on the various client devices,

wherein any of the layers may be changed without impacting other layers.

2. (Original) A server system as recited in claim 1, wherein the execution environment layer comprises a framework to receive the client requests and route the requests to the problem-solving logic for processing.

3. (Original) A server system as recited in claim 2, wherein the execution environment layer comprises one or more adapters to interface the framework with different types of the client devices.

4. (Original) A server system as recited in claim 1, wherein the execution environment layer comprises:

a model dispatcher to route the client requests to selected execution models in the problem-solving logic layer; and

a request dispatcher to structure the replies for return to the client devices.

5. (Original) A server system as recited in claim 1, wherein the multi-layer application can be adapted to receive requests from new client devices with incompatible communication protocols by substituting a new execution environment layer that supports the new client devices.

6. (Original) A server system as recited in claim 1, wherein one of the execution models is embodied as a set of discrete program modules, each program module performing a specific task.

7. (Original) A server system as recited in claim 1, wherein one of the execution models is embodied as an interaction-based model in which computer programs are defined by a series of interaction definitions.

8. (Original) A server system as recited in claim 1, wherein the execution models are embodied according to at least one of a command model, an action-view model, and a use case model.

9. (Original) A server system as recited in claim 1, wherein one of the execution models performs tasks according to a first business purpose, and the multi-layer application being reconfigurable to achieve a different business purpose by installing another execution model that performs tasks according to the second business purpose.

10. (Original) A server system as recited in claim 1, wherein the interfacing layer comprises:

a data abstraction layer to obtain data from the resources and map the data into a domain framework that models information flow for a specific problem domain; and

a data coordination layer that provides an interface for the problem-solving logic layer to access the domain framework of the data abstraction layer and obtain the data.

11. (Original) A server system as recited in claim 10, wherein the data coordination layer comprises one or more application data managers that interface the domain framework in the data abstraction layer into an application solution space of the problem-solving logic layer.

12. (Original) A server system as recited in claim 1, wherein the multi-layer application can be adapted to access new resources by substituting in a new interfacing layer that supports the new resources.

13. (Original) A server system as recited in claim 1, wherein the client devices support different data formats, the presentation layer being configured to select appropriate data formats for encoding the replies.

14. (Original) A server system as recited in claim 1, wherein the client devices support different communication protocols, the presentation layer being configured to select appropriate communication protocols for delivering the replies to the clients.

15. (Original) A server system as recited in claim 1, wherein the presentation layer is configured to determine how to display the replies for a particular client.

16. (Previously presented) A server system as recited in claim 1, wherein the presentation layer is configured to determine at least one of (1) a layout of individual replies, (2) display attributes in which to present the replies, and (3) a presentation theme.

17. (Original) A server system as recited in claim 1, wherein the presentation layer comprises:

a presentation module to determine how the replies will appear on the client devices to users; and

a rendering module, separate from the presentation module, to determine how to render the replies on the client devices.

18. (Original) A server system as recited in claim 1, further comprising an authentication module to authenticate the client devices or users of the client devices.

19. (Original) A server system as recited in claim 1, further comprising a constraint system to constrain operation of the multi-layer application according to a hierarchy of different constraints.

20. (Original) A server system as recited in claim 1, further comprising a constraint system to constrain operation of the multi-layer application according to multiple different constraints, the constraint system comprising a hierarchy of constraint layers, with each constraint layer containing a set of one or more constraints that customize operation of the multi-layer application.

21. (Original) A server system as recited in claim 1, further comprising:

a constraint hierarchy of multiple constraint layers, each constraint layer containing a set of one or more constraints that constrain operation of the multi-layer application, the constraint layers being organized within the constraint hierarchy such that a first constraint layer limits a second constraint layer but the second constraint layer does not limit the first constraint layer; and

a constraint resolver to resolve the constraint layers so that operation of the multi-layer application is constrained by a set of the constraints in the constraint layers.

22. (Original) A server system as recited in claim 21, wherein the hierarchy of constraints comprises constraints selected from a group of constraints comprising:

legally mandated constraints to constrain operation of the multi-layer application according to legal principles;

company-mandated constraints to constrain operation of the multi-layer application according to preferences of a company that operates the application;

customer constraints to constrain operation of the multi-layer application according to preferences of customers;

cultural constraints to constrain operation of the multi-layer application according to cultural aspects; and

end user constraints to constrain operation of the multi-layer application according to preferences of an end user.

23. (Original) A server system as recited in claim 1, further comprising a security policy enforcement module to enforce security restrictions on accessing information stored at the one or more resources.

24. (Original) A server system as recited in claim 23, wherein the security policy enforcement module is to enforce the security restrictions based on a set of low-level security rules defined using high-level permission concepts.

25. (Original) A server system as recited in claim 1, wherein the presentation layer includes a form processor to generate a data input form for the multi-layer application by automatically adding, to a form definition that defines the data input form, validation code to validate subsequent inputs to one or more fields of the data input form.

26. (Original) A server system as recited in claim 25, wherein the form processor is to generate the data input form by identifying one or more custom tags associated with the data

input form, to replace each of the one or more custom tags with another tag, and further to add to the form definition, for each of the one or more replaced tags, validation code to validate subsequent inputs to a field corresponding to the tag.

27. (Original) A server system as recited in claim 25, wherein the form processor is further to automatically identify one or more data input fields to be included in the form definition.

28. (Original) A server system as recited in claim 25, wherein the form processor is further to automatically identify one or more restrictions associated with a data input field of the data input form, and to determine the validation code based at least in part on the one or more restrictions.

29. (Original) A server system as recited in claim 1, further comprising:

a resource bundle containing locale-specific content that is authored for a particular locale; and

a resource bundle manager to populate a locale-independent core with the locale-sensitive content in the resource bundle to produce a computer-servable document that can be served by the multi-layer application to the particular locale.

30. (Original) A server system as recited in claim 29, wherein the resource bundle manager resides in the interfacing layer.

31. (Withdrawn) A business-oriented computer software architecture stored on one or more computer-readable media, comprising:

a framework module to receive client requests from different client devices;

a first business logic module to process the client requests received by the framework according to an associated business purpose, the first business logic module generating replies corresponding to the client requests;

a presentation module to structure the replies produced by the first business logic module in a manner that makes the replies presentable on the client devices; and

the business-oriented computer software architecture being reconfigurable to another business purpose by substituting a second business logic module for the first business logic module.

32. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, wherein the framework module forms a container for the first and second business logic modules that allows the first and second business logic modules to be selectively added or removed from the architecture.

33. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, wherein the first and second business logic modules are implemented as discrete programs, each program performing a specific task.

34. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, wherein the client devices support different data formats, the presentation layer being configured to select appropriate data formats for encoding the replies.

35. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, wherein the client devices support different communication protocols, the presentation layer being configured to select appropriate communication protocols for delivering the replies to the clients.

36. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, wherein the presentation module comprises:

a presentation component to structure how the replies will appear; and

a rendering component, separate from the presentation component, to configure how the replies are output on a particular client.

37. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, wherein the presentation module generates a data input form by automatically adding, to a form definition that defines the data input form, validation code to validate subsequent inputs to one or more fields of the data input form.

38. (Withdrawn) A business-oriented computer software architecture as recited in claim 37, wherein the presentation module automatically identifies one or more data input fields to be included in the form definition.

39. (Withdrawn) A business-oriented computer software architecture as recited in claim 37, wherein the presentation module automatically identifies one or more restrictions

associated with a data input field of the data input form and determines the validation code based at least in part on the one or more restrictions.

40. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, further comprising a domain framework to model information flow for a particular business domain.

41. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, further comprising a constraint hierarchy with a hierarchical set of constraints that specify how the replies should be structured to customize the replies.

42. (Withdrawn) A business-oriented computer software architecture as recited in claim 31, further comprising:

a resource bundle containing locale-specific content that is authored for a particular locale; and

a resource bundle manager to populate a locale-independent core with the locale-sensitive content in the resource bundle to produce the replies.

43. (Withdrawn) A business-oriented computer software architecture stored on one or more computer-readable media, comprising:

a first layer to obtain data from external resources and map the data into a domain framework that models information flow for a specific problem domain;

a second layer to provide an interface into the domain framework of the first layer;

a third layer to process requests and produce replies, the third layer utilizing the second layer to access the domain framework and retrieve the data obtained by the first layer from the external resources;

a fourth layer to structure the replies for presentation and rendering on diverse client devices; and

wherein anyone of the first layer, the second layer, the third layer, and the fourth layer may be changed independently of other layers to modify operation of the computer software architecture.

44. (Withdrawn) A business-oriented computer software architecture as recited in claim 43, wherein the first layer may be updated to obtain data from new external resources.

45. (Withdrawn) A business-oriented computer software architecture as recited in claim 43, wherein the third layer is an original third layer that processes the requests according to a first business domain, further comprising a new third layer substituted for the original third layer to process the requests according to a second business domain.

46. (Withdrawn) A business-oriented computer software architecture as recited in claim 43, wherein the fourth layer can be updated to structure the replies for new client devices.

47. (Withdrawn) A business-oriented computer software architecture as recited in claim 43, wherein the fourth layer can be modified to structure the replies for presentation to users in different regions of the world.

48. (Previously presented) A method for processing client requests in a system, comprising:

receiving requests from multiple clients, the requests being related to a business-related problem domain, wherein the business-related problem domain pertains to a particular category of business-related service;

processing the requests within problem-solving logic to produce replies within the business-related problem domain, the processing comprising requesting data to be used in formulating the replies;

retrieving the data from one or more external resources and mapping the data to a domain framework for the business-related problem domain, the domain framework being independent from the problem-solving logic; and

interfacing the problem-solving logic to the domain framework to obtain the data for use in processing the request,

wherein a new business-related problem domain can be exchanged for a previous business-related problem domain by replacing one or more components of the system, without having to reconstruct an entire application solution for the new business-related problem domain.

49. (Original) A method as recited in claim 48, further comprising structuring the replies for presentation to the clients.

50. (Original) A method as recited in claim 48, further comprising:

structuring the replies to define how the replies will appear when presented at the clients; and

independent of said structuring, conforming the replies to output capabilities of the clients.

51. (Original) A method as recited in claim 48, further comprising constraining how the replies are presented according to a hierarchy of constraints, wherein the hierarchy of constraints comprises multiple constraints such that a first constraint limits a second constraint but the second constraint does not limit the first constraint.

52. (Withdrawn) A method for creating server applications for multiple different problem domains, comprising:

providing a framework to receive client requests from multiple different clients, individual clients communicating messages using different protocols and formats;

providing a resource structure to access resources that provide data related to the different problem domains;

creating a first server application for a first problem domain by interfacing first domain logic with the framework and the resource structure, the first domain logic being configured to process the client requests for the first problem domain;

creating a second server application for a second problem domain by interfacing second domain logic with the framework and the resource structure, the second domain logic being configured to process the client requests for the second problem domain; and

the first and second domain logic producing replies that the framework returns to the clients using the protocols and messaging formats employed by the clients.

53. (Withdrawn) A method as recited in claim 52, wherein the providing a resource structure comprises providing a resource structure that retrieves data from one or more external resources and maps the data to a domain framework that models information flow for a specific problem domain.

54. (Withdrawn) A method as recited in claim 52, further comprising:

structuring the replies to define how the replies will appear when presented at the clients; and

independent of said structuring, conforming the replies to output capabilities of the clients.

55. (Withdrawn) A method as recited in claim 52, further comprising generating a data input form for service to a client by automatically adding, to a form definition that defines the data input form, validation code to validate subsequent inputs to one or more fields of the data input form.

56. (Withdrawn) A method as recited in claim 52, further comprising automatically identifying one or more data input fields to be included in a data input form based on the first and second domain logic.

57. (Withdrawn) A method as recited in claim 52, further comprising forming a reply for a particular locale by retrieving a locale-independent core that contains locale-independent elements and populating the locale-independent core with locale-sensitive content that is appropriate for the particular locale.

58. (Previously presented) A server system comprising:

one or more computers; and

a multi-layer application executing on the computers to handle client requests submitted by various client devices, the multi-layer application comprising:

a problem-solving logic layer to process the client requests according to an associated problem domain, wherein the problem domain pertains to a particular category of service, the problem-solving logic layer containing one or more execution models to perform various sets of tasks when processing the client requests, the problem-solving logic layer producing replies to the client requests;

an execution environment layer to receive the client requests and select an appropriate execution model in the problem-solving logic layer for processing the client requests;

an interfacing layer to interface the problem-solving logic layer with one or more resources so that the execution models may utilize the resources when processing the client requests,

wherein the interfacing layer comprises:

a data abstraction layer to obtain data from the resources and map the data into a domain framework that models information flow for a specific problem domain; and

a data coordination layer that provides an interface for the problem-solving logic layer to access the domain framework of the data abstraction layer and obtain the data; and

a presentation layer to receive the replies produced by the problem-solving logic layer and to structure the replies in a manner that makes the replies presentable on the various client devices.